Python – 8. Δομές Δεδομένων II

Συμβολοσειρές (strings)

Τα αλφαριθμητικά ή συμβολοσειρές στην Python είναι ακολουθίες από χαρακτήρες που έχουν σταθερό μέγεθος και μη μεταβαλλόμενα περιεχόμενα. Δηλαδή, δεν μπορούμε να προσθέσουμε ή να αφαιρέσουμε χαρακτήρες, ούτε να τροποποιήσουμε τα περιεχόμενα του αλφαριθμητικού. Γι' αυτό λέμε ότι η δομή αυτή ανήκει στις μη μεταβαλλόμενες (immutable) δομές της Python.

Η αρίθμηση των χαρακτήρων σε ένα αλφαριθμητικό ξεκινάει από το 0. Για παράδειγμα:

```
word = "PYTHON"
```

0	1	2	3	4	5
P	Y	T	H	O	N
word[0]	word[1]	word[2]	word[3]	word[4]	word[5]

Η συνάρτηση **len** επιστρέφει το μήκος, δηλαδή το πλήθος των χαρακτήρων του αλφαριθμητικού.

Παράδειγμα:
```
>>> word = "PYTHON"
>>> len(word)
6
```

Ο τελεστής + όταν εφαρμόζεται σε αντικείμενα τύπου string, έχει σαν αποτέλεσμα τη συνένωσή τους σε μια συμβολοσειρά.

Παράδειγμα:
```
>>> word = "PYTHON"
>>> print word[5]+word[0]
NP
```

Η συνάρτηση **str** μετατρέπει μια τιμή σε συμβολοσειρά.

Παράδειγμα:
```
>>> str(28) == "28"
True
```

Η συνάρτηση **int** μετατρέπει ένα αλφαριθμητικό στον ακέραιο αριθμό που αναπαριστά.

Παράδειγμα:
```
>>> print int("6") + 4
10
```

Έλεγχος ύπαρξης

Ο τελεστής **in** ελέγχει αν ένα αντικείμενο ανήκει σε ένα σύνολο αντικειμένων. Δεδομένου ότι οι συμβολοσειρές μπορούν να θεωρηθούν ως σύνολα χαρακτήρων, μπορούμε να τον χρησιμοποιήσουμε:
```
>>> "Py" in "Python"
True
>>> "a" in "Python"
False
>>> "a" not in "Python"
True
```

Οι **συγκριτικοί τελεστές** (<, <=, >, >=, ==) ισχύουν και στις συμβολοσειρές, η λειτουργία των οποίων βασίζεται στη λεξικογραφική διάταξη των χαρακτήρων.
```
>>> "Xara" > "Anna"
True
>>> "1000" < "8"
True
>>> "X5" > "X4"
True
```

Σάρωση χαρακτήρων και απαλοιφή κενών
```
def trimSpaces(sentence):
  result = ""
  for char in sentence :
    if char != " " :
      result += char
  return result

>>> phrase = "Houston we have a problem"
>>> trimSpaces(phrase)
'Houstonwehaveaproblem'
```

Σάρωση χαρακτήρων και έλεγχος ύπαρξης (καταμέτρηση φωνηέντων)
```
def countVowels(word):
  vowels = "AEIOUaeiou"
  count = 0
  for letter in word :
    if letter in vowels:
      count += 1
  return count

>>> phrase = "Houston we have a problem"
>>> countVowels(phrase)
9
```

Άσκηση 1η

Δημιουργήστε μια συνάρτηση που δέχεται ένα string και επιστρέφει το αντίστροφό του. Για παράδειγμα, δέχεται το "ΑΒΓ" και επιστρέφει το "ΓΒΑ". Με τη χρήση αυτής της συνάρτησης γράψτε πρόγραμμα το οποίο διαβάζει μία λέξη και ελέγχει αν είναι καρκινική ή όχι (διαβάζεται το ίδιο αν πάμε από το τέλος στην αρχή π.χ. "ANNA").
```
def InvStr(st):
  N=len(st)
  new=""
  for i in range(N-1,-1,-1):
    new = new+st[i]
  return new

s1=raw_input("Text:")
s2=InvStr(s1)
if s1==s2:
  print "καρκινική"
else:
  print "όχι καρκινική"
```

Άσκηση 2η

Δημιουργήστε μια συνάρτηση που δέχεται μία πρόταση και επιστρέφει το πλήθος των λέξεων που περιέχει.
Σημείωση: Μία πρόταση τελειώνει με τελεία (".") και κάθε λέξη διαχωρίζεται από την άλλη με ένα space (" ").

1η λύση
```
def LexeisSeProt(keimeno):
  diax=" ."
  lexeis=0
```

```
  for xar in keimeno:
    if xar in diax:
      lexeis += 1
  return lexeis

prot=raw_input("...")
print LexeisSeProt(prot)
```

2η λύση

```
def LexeisSeProt(keimeno):
  lexeis=0
  for xar in keimeno:
    if xar == " ":
      lexeis += 1
  lexeis += 1
  return lexeis

prot=raw_input("...")
print LexeisSeProt(prot)
```

Άσκηση 3η

Δημιουργήστε μια συνάρτηση που δέχεται ελληνικές φράσεις και μετρά το πλήθος των φωνηέντων που περιέχουν.

Σημείωση: Θεωρήστε ότι τα φωνήεντα είναι χωρίς τόνους και διαλυτικά.

```
def countGRVowels(word):
  vowels = "ΑΕΟΩΥΙΗαεοωυιηΑΕΟΩΥΙΗάέόώύίήΐΰϊϋΐΰ"
  count = 0
  for letter in word :
    if letter in vowels:
      count += 1
  return count

s=raw_input("Δώσε φράση στα Ελληνικά:")
print "Σε αυτήν, υπάρχουν",
countGRVowels(s),"φωνήεντα"
```

Άσκηση 4η

Δημιουργήστε μια συνάρτηση που δέχεται μία φράση και επιστρέφει τους 10 πρώτους χαρακτήρες της αντικαθιστώντας τυχόν κενά με την κάτω παύλα ("_").

Π.χ. Αν διαβάσει "ένα δύο τρία τέσσερα πέντε γάιδαροι" επιστρέφει "ένα_δύο_τρ".

Σημείωση: Αν η φράση έχει λιγότερους από 10 χαρακτήρες τότε επιστρέφει τόσους χαρακτήρες όσους δέχτηκε ...

```
def Put_(sentence):
  result = ""
  for char in sentence :
    if char != " " : result += char
    else: result += "_"
  return result

def trim10str(sentence):
  if len(sentence)<=10:
    result = sentence
  else:
    result=""
    for i in range(10):
      result += sentence[i]
  return result

s=raw_input("String: ")
s = Put_(s)
s = trim10str(s)
print s
```

Άσκηση 5η

Δημιουργήστε μια συνάρτηση που δέχεται μία φράση και την επιστρέφει αφαιρώντας τα κενά και κάνοντας κεφαλαίο το πρώτο γράμμα κάθε λέξης.

Σημείωση 1η: Αν διαβάσει "one two three four five donkeys" επιστρέφει "OneTwoThreeFourFiveDonkeys".

Σημείωση 2η: Χρησιμοποιείστε τη συνάρτηση ***upper()*** που επιστρέφει κεφαλαία, εφόσον οι χαρακτήρες είναι αγγλικοί! Αν s είναι ένα μη κενό string τότε το s.upper() το επιστρέφει με κεφαλαίους χαρακτήρες και το s[2].upper() επιστρέφει κεφαλαίο τον χαρακτήρα της θέσης 2.

```
def NewString(s):
  N = len(s)
  new = s[0].upper() #1ος χαρακτήρας πρέπει
να είναι κεφαλαίος
  for i in range(1,N):
    # αν προηγούμενος χαρακτήρας κενό,
τρέχον αντιγράφεται κεφαλαίος
    if s[i-1] == ' ': new += s[i].upper()
    # αν τρέχον χαρακτήρας μη κενό,
αντιγράφεται
    elif s[i] != ' ': new += s[i]
  return new
```

Άσκηση 6η

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει λέξεις από το πληκτρολόγιο και θα μετράει και θα εμφανίζει πόσες λέξεις ξεκινούν από το γράμμα Α (κεφαλαίο ή μικρό). Όταν δοθεί λέξη που να τελειώνει με το γράμμα Ω (κεφαλαίο ή μικρό) θα τερματίζει.

Σημείωση: δεν θα υπάρχει λέξη που να ξεκινά με το γράμμα Α και να τελειώνει με το γράμμα Ω.

```
alpha = 'αάΑA'
omega = 'ωώΩΩ'
lexeis_me_alpha = 0
lexi = raw_input('Δώσε λέξη:')
while lexi[len(lexi)-1] not in omega:
  if lexi[0] in alpha:
    lexeis_me_alpha += 1
  lexi = raw_input('Δώσε λέξη:')
print 'Λέξεις από "Α"', lexeis_me_alpha
```

ή

```
count=0
alpha="αάΑA"
omega="ωώΩΩ"
LexiOmega=False
st=raw_input("Δώσε λέξη: ")
while LexiOmega==False:
  L=len(st)
  if st[L-1] in omega:
    LexiOmega=True
  elif st[0] in alpha:
    count += 1
    st=raw_input("Δώσε λέξη: ")
  else:
```

```
    st=raw_input("Δώσε λέξη: ")
print count
```

Άσκηση 7η

Να γράψετε μια συνάρτηση η οποία θα ελέγχει αν μια συμβολοσειρά αποτελεί ηλεκτρονική διεύθυνση αλληλογραφίας ελληνικού ιστότοπου, δηλαδή:

- περιέχει το σύμβολο '@',
- δεν περιέχει κενά και
- έχει κατάληξη '.gr'.

```
def email(s):
  T = len(s)
  IsEmail = '@' in s
  if IsEmail: IsEmail = ' ' not in s
  if IsEmail: IsEmail = '.' == s[T-3]
  if IsEmail: IsEmail = 'g' == s[T-2]
  if IsEmail: IsEmail = 'r' == s[T-1]
  return IsEmail
```

ή

```
def email(s):
  T = len(s)
  if ('@' in s) and (' ' not in s) and ('.'
== s[T-3]) and ('g' == s[T-2]) and ('r' ==
s[T-1]):
    IsEmail = True
  else:
    IsEmail = False
  return IsEmail
```

Άσκηση 8η

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει λέξεις από το πληκτρολόγιο και θα τις ενώνει σε μια μεγάλη πρόταση, την οποία στη συνέχεια θα εμφανίζει στην οθόνη. Η εισαγωγή των λέξεων τερματίζεται όταν ως λέξη δοθεί η τελεία ".".

Σημείωση: Στην πρόταση οι λέξεις θα χωρίζονται μεταξύ τους με κενά και η πρόταση θα τελειώνει με τελεία "." .

```
protasi = ''
lexi=raw_input('Δώσε λέξη, "." για τέλος:')
while lexi != '.':
  protasi += lexi + ' '
  lexi=raw_input('Δώσε λέξη, "." για
τέλος:')
protasi += '.'
print protasi
```

ή

```
protasi = ''
FirstTime = True
lexi=raw_input('Δώσε λέξη, "." για τέλος:')
while lexi != '.':
  if FirstTime:
    protasi += lexi
    FirstTime = False
  else:
    protasi += ' ' + lexi
  lexi=raw_input('Δώσε λέξη, "." για
τέλος:')
protasi += '.'
print protasi
```

Λίστες (lists)

Είναι μια διατεταγμένη ακολουθία αντικειμένων, όχι απαραίτητα του ίδιου τύπου. Σε αντίθεση με το string, είναι μια **δυναμική δομή** στην οποία μπορούμε να προσθέτουμε ή να αφαιρούμε στοιχεία. Κάθε αντικείμενο της λίστας χαρακτηρίζεται από ένα μοναδικό αύξοντα αριθμό, ο οποίος ορίζει τη θέση του στη λίστα. Η προσπέλαση στα στοιχεία της λίστας γίνεται όπως στα string, με το όνομα της λίστας και τον αύξοντα αριθμό του αντικείμενου μέσα σε αγκύλες.

Η εντολή

```
L = [ 5, 3, 8, 21, 13, 34 ]
```

δημιουργεί τη μεταβλητή L που αναφέρεται

στη λίστα [5, 3, 8, 21, 13, 34]

0	1	2	3	4	5
5	3	8	21	13	34
L[0]	L[1]	L[2]	L[3]	L[4]	L[5]

Παράδειγμα

```
>>> DaysOfWeek =
["ΔΕΥ","ΤΡΙ","ΤΕΤ","ΠΕΜ","ΠΑΡ"]
>>> DaysOfWeek = DaysOfWeek + ["ΣΑΒ"]

>>> print DaysOfWeek # Εκδοση 2.7
['\xc4\xc5\xd5\xd4\xc5\xd1\xc1', '\xd4\xd1\
xc9\xd4\xc7', '\xd4\xc5\xd4\xc1\xd1\xd4\
xc7', '\xd0\xc5\xcc\xd0\xd4\xc7', '\xd0\
xc1\xd1\xc1\xd3\xca\xc5\xd5\xc7', '\xd3\
xc1\xc2\xc2\xc1\xd4\xcf']

>>> print DaysOfWeek # Εκδοση 3.4
['ΔΕΥ','ΤΡΙ','ΤΕΤ','ΠΕΜ','ΠΑΡ','ΣΑΒ']
```

Αν θέλουμε να προσθέσουμε ένα στοιχείο στο τέλος της λίστας γράφουμε:

```
Λίστα = Λίστα + [ στοιχείο ]
```

Αν θέλουμε να προσθέσουμε ένα στοιχείο στην αρχή της λίστας γράφουμε:

```
Λίστα = [ στοιχείο ] + Λίστα
```

Για τις λίστες:

- Δεν έχουν σταθερό μέγεθος, δηλαδή μπορούν να αυξάνονται και να μειώνονται κατά την εκτέλεση του προγράμματος.
- Η αρίθμηση των δεικτών ξεκινάει από το 0, όπως ακριβώς στα string (συμβολοσειρές).
- Είναι δυναμικές δομές και χαρακτηρίζονται από μεγάλη ευελιξία. Έτσι για παράδειγμα, μπορούμε να έχουμε σε μια λίστα ακόμα και στοιχεία διαφορετικού τύπου.

Στις λίστες μπορούμε να χρησιμοποιήσουμε

- τον τελεστή **in**,
- τη συνάρτηση **len** ή και
- τον τελεστή συνένωσης '+'

όπως ακριβώς και στα string (συμβολοσειρές).

Παράδειγμα

```
>>> fruits = ['apple','orange']
>>> len(fruits)
2
>>> 'apple' in fruits
True
>>> powers = [2, 4, 8, 16]
>>> powers + fruits
[2,4,8,16,'apple','orange']
>>> powers = [2, 4, 8, 16]
>>> fib = [3, 5, 8, 13, 21]
>>> fib + powers
[3, 5, 8, 13, 21, 2, 4, 8, 16]
>>> powers = powers + [ powers[3] * 2 ]
>>> print powers
[2, 4, 8, 16, 32]
```

item in List

επιστρέφει True, αν το στοιχείο item υπάρχει μέσα στη λίστα List, αλλιώς επιστρέφει False.

item not in List

επιστρέφει True, αν το στοιχείο item δεν υπάρχει μέσα στη λίστα List, αλλιώς, αν υπάρχει επιστρέφει False.

len (List)

Επιστρέφει το πλήθος των στοιχείων (ή μέγεθος) της λίστας.

list (String)

Επιστρέφει μια λίστα με στοιχεία τους χαρακτήρες της συμβολοσειράς string. Η συνάρτηση αυτή μπορεί να μετατρέψει και άλλα είδη δομών σε λίστα.

Μέθοδοι

Οι λίστες διαθέτουν μεγάλη ποικιλία μεθόδων:

L.append(object)

προσθήκη του στοιχείου object στο τέλος της λίστας L

L.insert(index, object)

προσθήκη του στοιχείου object, στη θέση index της λίστας L

L.pop(index)

αφαίρεση από τη λίστα L του στοιχείου που βρίσκεται στη θέση index. Αν δεν δοθεί θέση, τότε θα αφαιρεθεί το τελευταίο στοιχείο της λίστας. Η pop επιστρέφει τιμή!

Διάσχιση Λίστας

Μπορούμε να επεξεργαστούμε τα στοιχεία μιας λίστας, ένα κάθε φορά, κάνοντας χρήση του παρακάτω ιδιώματος της δομής επανάληψης for:

```
for item in List :
  Εντολές Επεξεργασίας του αντικειμένου item
```

Δημιουργία και εμφάνιση στοιχείων λίστας

Οι παρακάτω εντολές αρχικά κατασκευάζουν μια λίστα η οποία περιέχει όλους τους αριθμούς από το 1 έως και το 6 και στη συνέχεια εμφανίζουν κάθε αριθμό σε διαφορετική γραμμή.

```
L = [1, 2, 3, 4, 5, 6]
for number in L :
  print number
```

Μέσος όρος των στοιχείων μιας λίστας

Για να υπολογίσουμε το μέσο όρο των στοιχείων μιας λίστας, πρώτα χρειάζεται να υπολογίσουμε το άθροισμα των στοιχείων, χρησιμοποιώντας μια μεταβλητή στην οποία προσθέτουμε, κάθε φορά, το επόμενο στοιχείο της λίστας:

```
sum = 0.0
for number in L :
  sum = sum + number
average = sum / len( L )
print average
```

Μέγιστη τιμή σε μια λίστα

```
maximum = L[0]
for number in L :
  if number > maximum :
    maximum = number
print maximum
```

Παράδειγμα: Ρέστα [Δ]

Καλούμαστε να σχεδιάσουμε το λογισμικό μιας ταμειακής μηχανής, το οποίο θα διαβάζει το ποσό που έδωσε ο πελάτης και το κόστος των αγορών του και θα εμφανίζει το πλήθος των κερμάτων ή χαρτονομισμάτων που θα δοθούν για ρέστα. Θεωρήστε ότι όλες οι τιμές είναι σε ακέραια πολλαπλάσια του ευρώ.

```
nomismata = [100, 50, 20, 10, 5, 2, 1]
cost = input( "Κόστος αγορών:    " )
payment = input( "Ποσό της πληρωμής:" )
change = payment - cost
counter = 0
for value in nomismata :
  counter = counter + ( change // value )
  change = change % value
print counter
```

Διαχωρισμός λίστας αριθμών σε ΘΕΤΙΚΟΥΣ, ΑΡΝΗΤΙΚΟΥΣ

```
positives = negatives = [ ]
for stoixeio in Lista :
  if stoixeio > 0 :
    positives = positives + [ stoixeio ]
  elif stoixeio < 0 :
    negatives = negatives + [ stoixeio ]
print positives, negatives
```

Συγχώνευση διατεταγμένων λιστών

```
def merge( A, B ) :
  L = [ ]
  while A != [ ] and B != [ ] :
    if A[0] < B[0] :
      L = L + [ A.pop(0) ]
    else :
      L = L + [ B.pop(0) ]
  return L + A + B
```

Η συνάρτηση range

Παραδείγματα

```
>>> range(4)
```

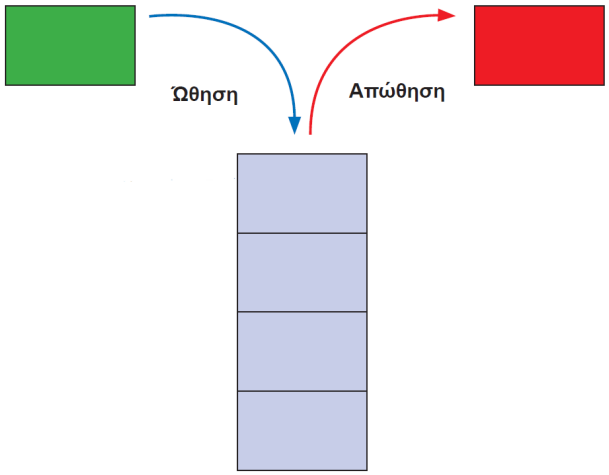
```
[0, 1, 2, 3]
>>> range(0,4)
[0, 1, 2, 3]
>>> range(0,4,1)
[0, 1, 2, 3]
>>> range(10,30,5)
[10, 15, 20, 25]
>>> range(30,10,-5)
[30, 25, 20, 15]
>>> range(1,2,100)
[1]
>>> range(1, 1, 100)
[ ]
>>> range(1, 5, -1)
[ ]
>>> range(5, 1, 1)
[ ]
>>> range(0)
[ ]
```

Τα παρακάτω τμήματα κώδικα εκτελούν την ίδια λειτουργία:

```
>>> L = [ 6, 28, 496, 8128 ]
>>> for item in L: print item
>>> for index in range(0,4): print L[index]
>>> for index in [0,1,2,3]: print L[index]
6
28
496
8128
```

Στοίβα (stack)

Last In First Out - LIFO



Όταν η στοίβα είναι άδεια, είναι προφανές ότι δεν μπορεί να γίνει απώθηση. Άρα, όταν απωθούμε ένα στοιχείο από τη στοίβα, θα πρέπει προηγουμένως να έχουμε εξασφαλίσει ότι η στοίβα δεν είναι κενή. Για αυτό το λόγο, εκτός από την ώθηση και την απώθηση, πρέπει να υλοποιήσουμε και τον έλεγχο, αν η στοίβα είναι κενή.

Οι βασικές λειτουργίες που πρέπει να υποστηρίζει η υλοποίηση μιας στοίβας είναι:

- **Δημιουργία μιας κενής στοίβας**.
- **Έλεγχος, αν η στοίβα είναι κενή**.
- **Ώθηση** ενός στοιχείου στη στοίβα.
- **Απώθηση** ενός στοιχείου από τη στοίβα.

```
def push(st,item):
  st.insert(0, item)

def pop(st):
  return st.pop(0)

def isEmpty(st):
  return len(st) == 0

def createStack():
  return []
```

Αντιστροφή αριθμών

Έστω πρόγραμμα που δέχεται από το χρήστη αριθμούς μέχρι να δοθεί το 0 και τους εμφανίζει σε αντίστροφη σειρά από αυτή με την οποία δόθηκαν.

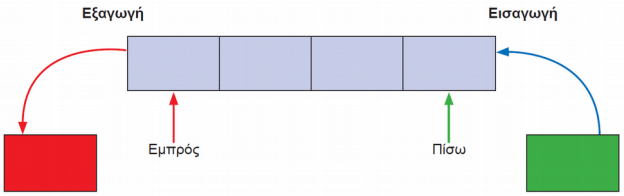
Θέλουμε κάθε φορά να εμφανίσουμε πρώτο τον αριθμό που δόθηκε τελευταίος.

Χρειαζόμαστε μια δομή δεδομένων που να υποστηρίζει τη λειτουργία LIFO, όπως η στοίβα.

```
S = createStack()
num = input()
while num != 0:
  push(S,num)
  num = input()
while not isEmpty(S):
  num = pop(S)
  print num
```

Ουρά (queue)

First In First Out - FIFO



Δύο είναι οι βασικές λειτουργίες μιας ουράς:

- **Εισαγωγή** στοιχείου, η οποία γίνεται στο πίσω μέρος της ουράς.
- **Εξαγωγή** στοιχείου, η οποία γίνεται από το εμπρός μέρος της ουράς.

Οι βασικές λειτουργίες που πρέπει να υποστηρίζει η υλοποίηση μιας ουράς είναι:

- **Δημιουργία** μιας κενής ουράς.
- **Έλεγχος**, αν η ουρά είναι κενή.
- **Εισαγωγή** στοιχείου.
- **Εξαγωγή** στοιχείου.

```
def enqueue(q,item):
  q.append(item)

def dequeue(q):
  return q.pop(0)
```

```
def isEmpty(q):
  return len(q) == 0

def createQueue():
  return []
```

Ασκήσεις με λίστες

Δραστηριότητα 7 [Ε]

Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίστα από λέξεις και θα επιστρέφει τη λέξη με το μεγαλύτερο μήκος.

```
def maxWord(L):
  max_str=L[0]
  max_len=len(max_str)
  for i in range(len(L)):
    if len(L[i]) > max_len:
      max_str = L[i]
      max_len = len(max_str)
  return max_str
Lista=["123","123456789","12345"]
print maxWord(Lista)
```

ΛΙΣΤΕΣ: Εισαγωγή Στοιχείων

Άσκηση 1η

Δημιουργήστε μια λίστα με τους βαθμούς στο μάθημα του Προγραμματισμού των 20 μαθητών της τάξης.

Λύση

```
L = []
for i in range(20):
  v = input("Δώστε βαθμό: ")
  L += [v]
```

Άσκηση 2η

Δημιουργήστε μια λίστα με τα ονόματα των μαθητών της τάξης. Η εισαγωγή ονομάτων σταματά όταν δοθεί το όνομα “ΤΕΛΟΣ”.

Λύση

```
N = []
name = raw_input("Δώσε όνομα: ")
while name != "ΤΕΛΟΣ":
  N += [name]
  name = raw_input("Δώσε όνομα: ")
```

ΛΙΣΤΕΣ: Εύρεση

Άσκηση 3η

Έστω λίστα με τους βαθμούς των μαθητών της τάξης στο μάθημα του Προγραμματισμού. Να γραφεί συνάρτηση η οποία θα δέχεται τη λίστα και θα επιστρέφει τον μεγαλύτερο από από αυτούς.

Λύση 1η

```
def maxList(L):
  N = len(L)
  maxL = L[0]
  for i in range(1,N):
    if L[i] > maxL:
      maxL = L[i]
  return maxL
```

Παρατηρήσεις

1. Παρατηρήστε ότι η παραπάνω συνάρτηση επιστρέφει το μέγιστο στοιχείο οποιασδήποτε λίστας έχει διατεταγμένα στοιχεία (δηλαδή στοιχεία στα οποία μπορεί να χρησιμοποιηθεί ο τελεστής **>**).

2. Για τη συγκεκριμένη άσκηση θα μπορούσε να είχε δοθεί αρχική τιμή ίση με μηδέν: **maxL = 0**.

3. Αν η **for** γραφόταν **for i in range(N)** δεν θα ήταν λάθος.

Λύση 2η

```
def maxList(L):
  maxL = L[0]
  for stoixeio in L:
    if stoixeio > maxL:
      maxL = stoixeio
  return maxL
```

Σημείωση

Για να ελέγξουμε τα παραπάνω γράφουμε π.χ.:

```
Lista = [12,10.5,15,18,9,6,19.5]
def maxList(L):
  ...
print maxList(Lista)
```

Άσκηση 4η

Έστω λίστα με τους βαθμούς των μαθητών της τάξης στο μάθημα του Προγραμματισμού. Να γραφεί συνάρτηση η οποία θα δέχεται τη λίστα και θα επιστρέφει τη θέση μέσα στη λίστα στην οποία βρίσκεται ο μεγαλύτερος.

Σημείωση: Θεωρούμε ότι όλοι οι βαθμοί είναι διαφορετικοί μεταξύ τους και άρα, ο μεγαλύτερος είναι ένας και άρα... η θέση στην οποία βρίσκεται μοναδική!

Λύση

```
def maxListPos(L):
  N = len(L)
  maxL = L[0]
  maxPos = 0
  for i in range(1,N):
    if L[i] > maxL:
      maxL = L[i]
      maxPos = i
  return maxPos
```

ΛΙΣΤΕΣ: Επεξεργασία

Άσκηση 5η

Έστω λίστα με τους βαθμούς των μαθητών της τάξης στο μάθημα του Προγραμματισμού. Να γραφεί συνάρτηση η οποία θα επιστρέφει τον μέσο όρο τους.

Λύση

```
def ListAverage(L):
  S = 0.0
  for stoixeio in L:
    S += stoixeio
  return S / len(L)
```

Άσκηση 6η

Έστω λίστα με τους βαθμούς των μαθητών της τάξης στο μάθημα του Προγραμματισμού. Να γραφεί συνάρτηση η οποία θα επιστρέφει το πλήθος όσων είναι κάτω από τη βάση (10).

Λύση
```
def KatoApo10(L):
  n = 0
  for stoixeio in L:
    if stoixeio < 10:
      n += 1
  return n
```

Άσκηση 7η

Να γραφεί πρόγραμμα το οποίο:

1. Διαβάζει τα ονόματα και τον τελικό βαθμό των 10 μαθητών της τάξης.

2. Υπολογίζει και εμφανίζει το μέσο όρο της βαθμολογίας τους.

3. Εμφανίζει τα ονόματα των μαθητών που έχουν τελικό βαθμό κάτω από 10.

Λύση 1η
```
N = 25
O = []
V = []
S = 0.0
for i in range(N):
  onoma = raw_input("Ονομα:")
  vathmos = input("Βαθμός:")
  O += [onoma]
  V += [vathmos]
  S += vathmos
mo = S / N
print "Μέσος Όρος:", mo
print "Μαθητές κάτω από βάση"
for i in range(N):
  if V[i] < 10:
    print O[i]
```

Λύση 2η
```
N = 25
O = []
V = []
Kato10 = []
S = 0.0
for i in range(N):
  onoma = raw_input("Ονομα:")
  vathmos = input("Βαθμός:")
  O += [onoma]
  V += [vathmos]
  S += vathmos
  if vathmos < 10:
    Kato10 += [onoma]
mo = S / N
print "Μέσος Όρος:", mo
print "Μαθητές κάτω από βάση"
for stoixeio in Kato10:
  print stoixeio
```

Άσκηση 8η

Να γραφεί πρόγραμμα το οποίο:

1. Διαβάζει τα ονόματα και τις μέγιστες θερμοκρασίες 20 πόλεων της Ευρώπης.

2. Υπολογίζει και εμφανίζει το μέσο όρο της θερμοκρασίας στις πόλεις αυτές.

3. Εμφανίζει το πλήθος και τα ονόματα των πόλεων όπου η θερμοκρασία ήταν πάνω από 20^{o} C.

Λύση
```
...
```

ΛΙΣΤΕΣ: Ταξινόμηση

Συνάρτηση Ταξινόμησης Ευθείας Ανταλλαγής ή Ταξινόμησης Φυσαλίδας (Bubble Short)

Η ταξινόμηση που γίνεται είναι **αύξουσα**.

```
def BubbleSort(L):
  N = len(L)
  for i in range(1,N,1):
    for j in range(N-1,i-1,-1):
      if L[j] < L[j-1]:
        L[j],L[j-1] = L[j-1],L[j]
  return L
```

Άσκηση 11η

Να γραφεί πρόγραμμα σε python το οποίο διαβάζει τα ονόματα και τον τελικό μέσο όρο της βαθμολογίας των 25 μαθητών μιας τάξης και εμφανίζει με φθίνουσα σειρά βαθμολογίας τα ονόματα και το βαθμό των μαθητών που έχουν τελικό μέσο όρο πάνω από 18. Σε περίπτωση ισοβαθμίας τα στοιχεία να εμφανίζονται με αλφαβητική σειρά.

Λύση
```
N = 25
for i in range(N):
  O[i] = raw_input('Όνομα: ')
  V[i] = input('Βαθμός: ')

for i in range(1,N,1):
  for j in range(N-1,i-1,-1):
    if V[j] > V[j-1]:
      V[j],V[j-1] = V[j-1],V[j]
      O[j],O[j-1] = O[j-1],O[j]
    elif V[j] == V[j-1] and
         O[j] < O[j-1]:
      O[j],O[j-1] = O[j-1],O[j]

i = 0
while V[i] > 18:
  print O[i], V[i]
  i += 1
```